Fast Structural Alignment of RNAs by Optimizing the Adjoining Order of Profile-csHMMs

Byung-Jun Yoon, *Member, IEEE*, and P. P. Vaidyanathan, *Fellow, IEEE*

***Abstract*—Recently, a novel RNA structural alignment method has been proposed based on profile-csHMMs. In principle, the profile-csHMM based approach can handle any kind of RNA secondary structures including pseudoknots, and it has been shown that the proposed approach can find highly accurate RNA alignments. In order to find the optimal alignment, the method employs the SCA algorithm that can be used for finding the optimal state sequence of profile-csHMMs. The computational complexity of the SCA algorithm is not fixed, and it depends on the so-called adjoining order that describes how we can trace-back the optimal state sequence in a given profile-csHMM. Therefore, for fast RNA structural alignments, it is important to find the adjoining order that has the minimum computational cost. In this paper, we propose an efficient algorithm that can systematically find the optimal adjoining order that minimizes the computational cost for finding the RNA alignments. Numerical experiments show that employing the proposed algorithm can make the alignment speed up to 3.6 times faster, without any degradation in the quality of the RNA alignments.**

***Index Terms*—Profile-csHMM, pseudoknot, RNA homology search, RNA structural alignment, SCA algorithm.**

I. Introduction

MANY functional RNAs can fold onto themselves to form base-paired secondary structures. It is known that these structures are crucial for carrying out the functions of the RNAs and, therefore, the secondary structure is typically conserved among homologous RNAs. For this reason, when carrying out a similarity search for ncRNAs, it is advantageous to consider both the structural similarity and the sequence similarity [1], [13]. A conserved secondary structure that is shared by homologous RNAs manifests itself in pairwise correlations between distant bases in each of these RNA sequences. Therefore, in order to perform an accurate RNA similarity search, we need a statistical model that can describe such base correlations and can also be used to recognize them.

Until now, various models have been used in RNA sequence analysis, where examples include CMs (covariance models) [2], PHMMTSs (pair hidden Markov models on tree structures) [9], and PSTAGs (pair stochastic tree adjoining grammars) [6], just to name a few. These models provide effective frameworks for representing RNAs with a conserved secondary structure and for finding their homologues. Each of these models can handle a limited class of RNA secondary structures that depends on its descriptive capability. For example, the CMs [2] and the PHMMTSs [9] are both based on SCFGs (stochastic context-free grammars), and since SCFGs cannot describe crossing correlations, these models are incapable of handling *pseudoknots*.[1] PSTAGs [6], which employ special subclasses of tree adjoining grammars (TAGs), are a recent development that can also handle simple pseudoknots that include many known pseudoknots, but not all of them.

Recently, we have proposed a new statistical model called profile-csHMMs (profile context-sensitive hidden Markov models) that can represent any kind of pairwise base correlations [12], [15]. Consequently, profile-csHMMs can in principle handle any kind of pseudoknots, which is an important advantage over many existing models. Profile-csHMMs have been applied to the problem of finding structural alignments of RNAs [15], and it was shown that the profile-csHMM based approach can find highly accurate alignments of RNAs that have complex secondary structures. Like many other statistical models used in RNA analysis, one practical problem in using profile-csHMMs is its high computational cost. In order to deal with complicated base correlations, the SCA (sequential component adjoining) algorithm [12] that is used for finding the optimal state sequence of profile-csHMMs makes important generalizations to existing algorithms such as the Viterbi algorithm and the Cocke–Younger–Kasami (CYK) algorithm [1]. As a result, the SCA algorithm has a variable computational complexity that depends on the so-called "adjoining order" that describes how the sequence adjoining rules should be applied to find the optimal state sequence in a given profile-csHMM. Therefore, in order to make the structural alignment based on profile-csHMMs faster, we have to find the adjoining order that minimizes the computational cost for finding the optimal alignment.

In this paper, we propose an efficient algorithm that can be used for finding the optimal adjoining order for a fast structural alignment of RNAs. The paper is organized as follows. In Section II, we briefly review the structural alignment method based on profile-csHMMs. In Section III, we consider the problem of optimizing the adjoining order in more depth. The algorithm for finding the optimal adjoining order is described in Section IV. Finally, we demonstrate the performance of the proposed method in Section V and conclude the paper in Section VI.

[1]RNA secondary structures that have crossing base-pairs are called pseudoknots.

Manuscript received September 2, 2007; revised March 8, 2008. This work was supported in part by the National Science Foundation under Grant CCF-0636799. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ahmed H. Tewfik.

B.-J. Yoon is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3128 USA (e-mail: bjyoon@ece.tamu.edu).

P. P. Vaidyanathan is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA.

Digital Object Identifier 10.1109/JSTSP.2008.923846

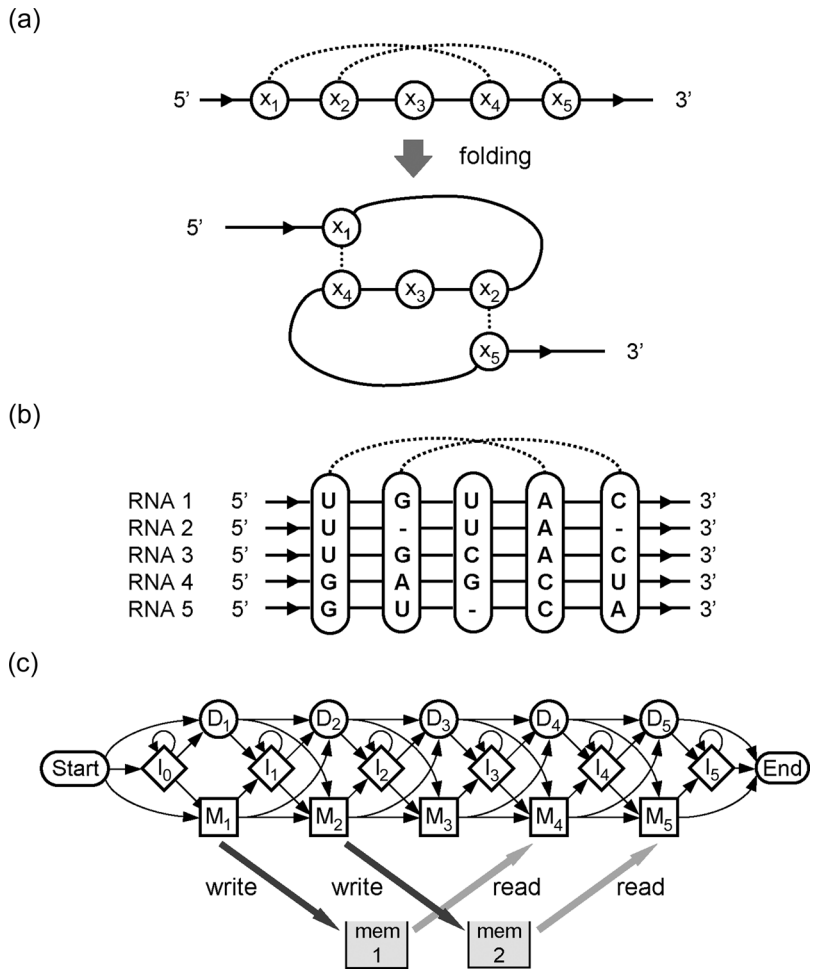


Fig. 1. (a) Reference RNA sequence and its secondary structure after folding. (b) RNA multiple sequence alignment. (c) Profile-csHMM that is constructed based on the reference RNA.

II. Review of RNA Structural Alignment Using Profile Context-Sensitive HMMs

RNA structural alignment methods try to find the optimal alignment between a structured RNA (an RNA with a known secondary structure) and an unstructured RNA (an RNA with an unknown structure) [6], [15]. Unlike traditional sequence alignment methods [4], structural alignment methods consider both structural similarity as well as sequence similarity between the RNAs to find the best alignment between them. As many functional RNAs conserve their secondary structure more than they conserve their primary sequence, structural alignment methods can find more accurate RNA alignments than methods that are solely based on sequence similarity. Furthermore, when applied to RNA similarity search, the structural alignment approach can significantly increase the discriminative power of the search, as it can detect homologues with poor sequence conservation and also reject false candidates that lack the conserved secondary structure.

Recently, we have proposed a new RNA structural alignment method based on profile-csHMMs [15]. The profile-csHMM is a subclass of the context-sensitive HMM (csHMM) [11], which has a linear repetitive structure that is suitable for representing RNA sequence profiles. As profile-csHMMs can represent any kind of pairwise base correlations, they provide a convenient framework for building RNA sequence analysis tools that can virtually handle any kind of RNA secondary structures including pseudoknots.

A. Building the Model

Assume that we want to build a profile-csHMM that represents the reference RNA shown in Fig. 1(a). The dotted lines indicate the base-pairs. The reference RNA can be either a single RNA sequence or the consensus sequence of an alignment of related RNAs as shown in Fig. 1(b). Fig. 1(c) shows the profile-csHMM that has been constructed based on the reference RNA. As we can see in Fig. 1(c), the profile-csHMM has a very regular structure that repetitively uses *match states* (M_k), *delete states* (D_k), and *insert states* (I_k). The match state M_k is used to represent the kth base in the reference RNA, while the delete state D_k is used to model its deletion. The insert state I_k is used to model additional bases that are inserted between the kth and the $(k+1)$th bases of the original RNA. One important feature of the profile-csHMM is the context-sensitivity of certain match states. In order to model the pairwise correlation between two bases, the profile-csHMM uses a pair of *pairwise-emission* match state and *context-sensitive* match state, which cooperate

with each other as follows. When we enter a pairwise-emission state, it emits a symbol according to its emission probabilities. After emitting a symbol, the pairwise-emission match state stores the emitted symbol in the auxiliary memory that is dedicated to it. Later on, when we enter the corresponding context-sensitive match state, it first reads the symbol that was previously emitted at the pairwise-emission state and then its emission probabilities are adjusted accordingly.

For example, let us take a look at the profile-csHMM in Fig. 1(c). In order to model the base-pair between the first and the fourth bases in the original RNA, we use a pairwise-emission match state at M_1 and the corresponding context-sensitive match state at M_4. Let us assume that M_1 emitted the base A, which will be subsequently stored in MEMORY 1. When we enter M_4, it first reads from MEMORY 1 to see that A was emitted at M_1. Based on this observation, the emission probabilities at M_4 are adjusted such that M_4 emits the base U (which is the complementary base of A). For other bases, the model works in a similar manner to maintain the base-pair in the original RNA. Similarly, the base correlation between the second and the fifth bases in the reference RNA are modeled by the pair of pairwise-emission match state at M_2 and the corresponding context-sensitive match state at M_5 that share MEMORY 2. Further details on the concept of profile-csHMMs and their construction can be found in Section IV of [15].

B. Finding the Structural Alignment

Once we have constructed the profile-csHMM that represents the reference RNA, we can use it to find the structural alignment between the reference RNA and an unstructured target RNA. The optimal alignment between these RNAs can be obtained by finding the optimal state sequence of the profile-csHMM that maximizes the observation probability of the target RNA. As every match state in the model corresponds to a specific base position in the reference RNA, finding the underlying state sequence of the target RNA leads to a unique and unequivocal alignment between the RNAs.

As profile-csHMMs can describe complicated symbol correlations that cannot be modeled using HMMs or SCFGs, we can neither use the Viterbi algorithm (used for HMMs) nor the CYK algorithm (used for SCFGs) to find the optimal state sequence (or, optimal path) in profile-csHMMs. For this reason, a new dynamic programming algorithm, called the SCA (sequential component adjoining) algorithm, has been proposed for finding the optimal state sequence in profile-csHMMs [12], [15]. The SCA algorithm makes two important generalizations to the aforementioned algorithms. First, it allows us to define and use subsequences that consist of multiple nonoverlapping intervals. This tremendously increases the number of ways in which the intermediate subsequences (used during the iterative process of finding the optimal state sequence) can be adjoined and extended, and it allows us to handle symbol correlations that are intertwined in a complicated manner. Second, the SCA algorithm follows a model-dependent "adjoining order" to find the optimal state sequence, instead of simply proceeding left-to-right (like the Viterbi algorithm) or inside-to-outside (like the CYK algorithm). Like these two algorithms, the SCA algorithm first finds the optimal paths of short subsequences, and iteratively extends or adjoins them to find the optimal paths of longer subsequences. This process is repeated until we find the optimal path for the entire symbol sequence. During this iterative process, the adjoining order instructs the algorithm how to proceed in extending and adjoining the subsequences so that we can ultimately find the final optimal state sequence.

Usually, it is convenient to describe the *adjoining order* in terms of the bases and base-pairs in the reference RNA that was used to construct the profile-csHMM. The adjoining order shows how the bases can be "assembled" step-by-step, to obtain the entire RNA sequence. This is illustrated on the left-hand side of Fig. 2. According to this order, we progressively find the optimal state sequence for the portion of the profile-csHMM that corresponds to the partial RNA shown on the left-hand side.[2] This is shown on the right-hand side of Fig. 2. By following the adjoining order, we can ultimately find the optimal state sequence of the observed symbol sequence (i.e., the target RNA sequence), hence the best structural alignment between the reference RNA and the target RNA.

III. Considerations for Finding a Good Adjoining Order

In the previous section, we briefly reviewed the RNA structural alignment method based on profile-csHMMs. As we have seen, we first construct a profile-csHMM that represents the given reference RNA and then use the SCA algorithm to find the optimal state sequence of the target RNA. Based on the predicted optimal state sequence, we can immediately find the optimal alignment between the reference and the target RNAs. Now, one question that is of practical significance is the following: Given a reference RNA with a known structure, how can we find a "good" adjoining order that can be used to find the optimal state sequence of the corresponding profile-csHMM? This is an important question as the choice of the adjoining order will affect the quality of the RNA alignment as well as the computational cost for finding the alignment. To answer this question, we address two important issues that need to be considered in finding a good adjoining order.

A. Definition of Adjoining Rules

Before proceeding with the discussion, let us first define the concept of adjoining rules. As the name suggests, an *adjoining rule* describes how we can adjoin two nonoverlapping subsequences to obtain a longer subsequence. Examples of adjoining rules are shown in Fig. 3. For example, let us consider the adjoining rule shown on the left-hand side of Fig. 3(a). This rule shows how we can adjoin a subsequence with two intervals (sequence a) and a subsequence with a single base (sequence b) to obtain a new subsequence that has also two intervals (sequence c). Similarly, the rule illustrated in Fig. 3(b) shows how we can adjoin a subsequence with two intervals (sequence a) and another subsequence with two intervals that consists of a single

[2]In practice, we only compute the maximum observation probability of the subsequence, and the optimal state sequence that gives rise to the maximum probability is traced back later, once we have computed the maximum probability of the entire symbol sequence.

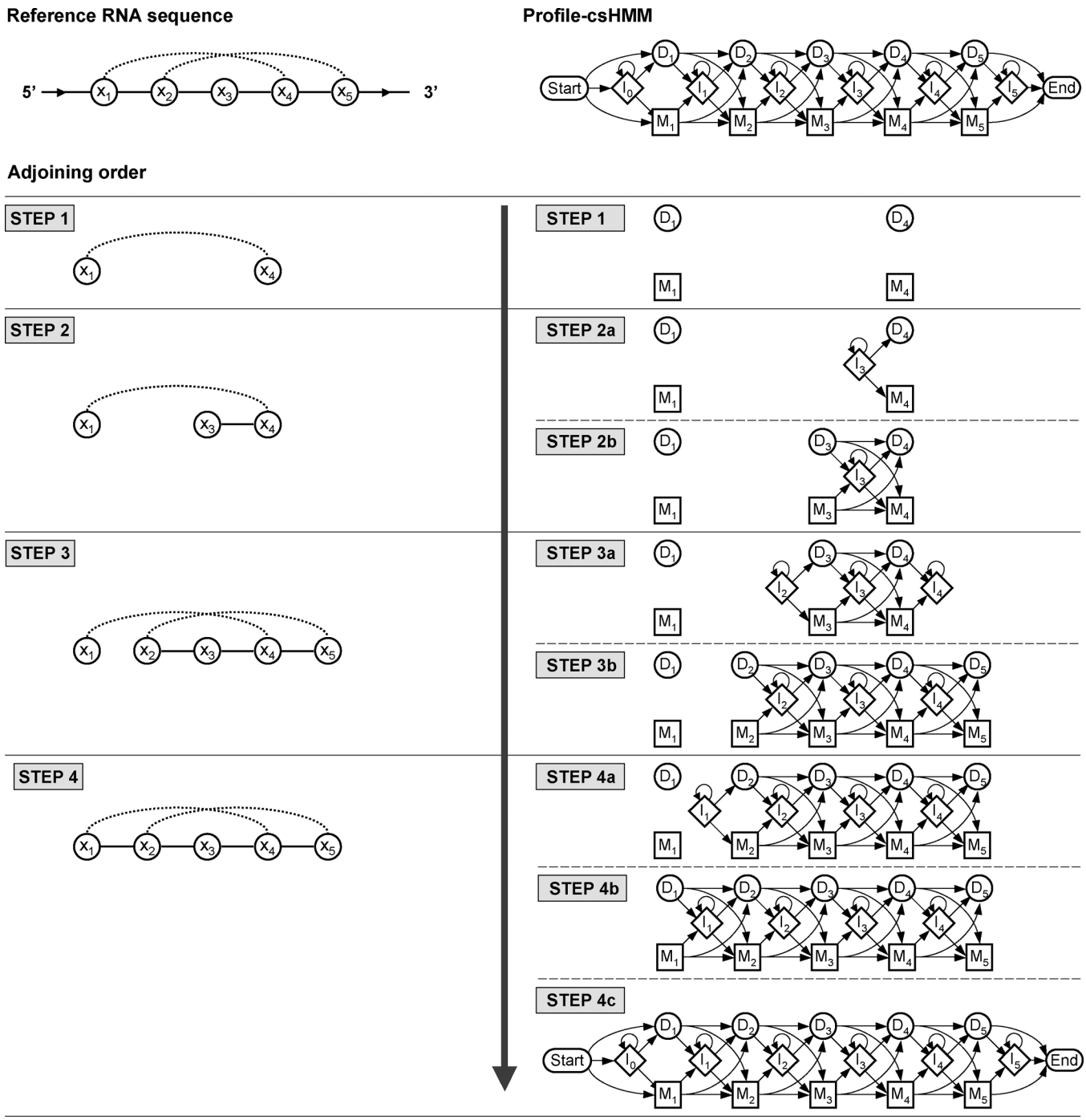


Fig. 2. Example of a possible adjoining order.

base-pair (sequence b) to get a longer subsequence with two intervals (sequence c). Another rule depicted in Fig. 3(c) shows how we can adjoin two neighboring subsequences each with a single interval (sequences a and b) to obtain a new sequence that has only one interval (sequence c). Let us consider the adjoining order shown in Fig. 2 (left-hand side). We can obtain the partial RNA in STEP-2 by applying the adjoining rule in Fig. 3(a). Similarly, we can obtain the partial RNAs shown in STEP-3 and STEP-4 by using the rules in Fig. 3(b) and (c), respectively.

The rules for adjoining "symbol sequences" (shown on the left-hand side of Fig. 3) are ultimately used for adjoining the "optimal state sequences" (shown on the right-hand side of Fig. 3) that correspond to certain portions of the profile-csHMM. As an example, let us again focus on the adjoining rule in Fig. 3(a). Consider a subsequence with two intervals, where the first interval consists of only one symbol at position j and the second interval is located between positions k and m. Assume that the underlying state at position j is M_1 and the states at positions k and m are M_3 and M_4, respectively. How can we find the optimal state sequence of the given subsequence by combining shorter optimal state sequences? As shown on the right-hand side of Fig. 3(a), we can adjoin the *optimal state*

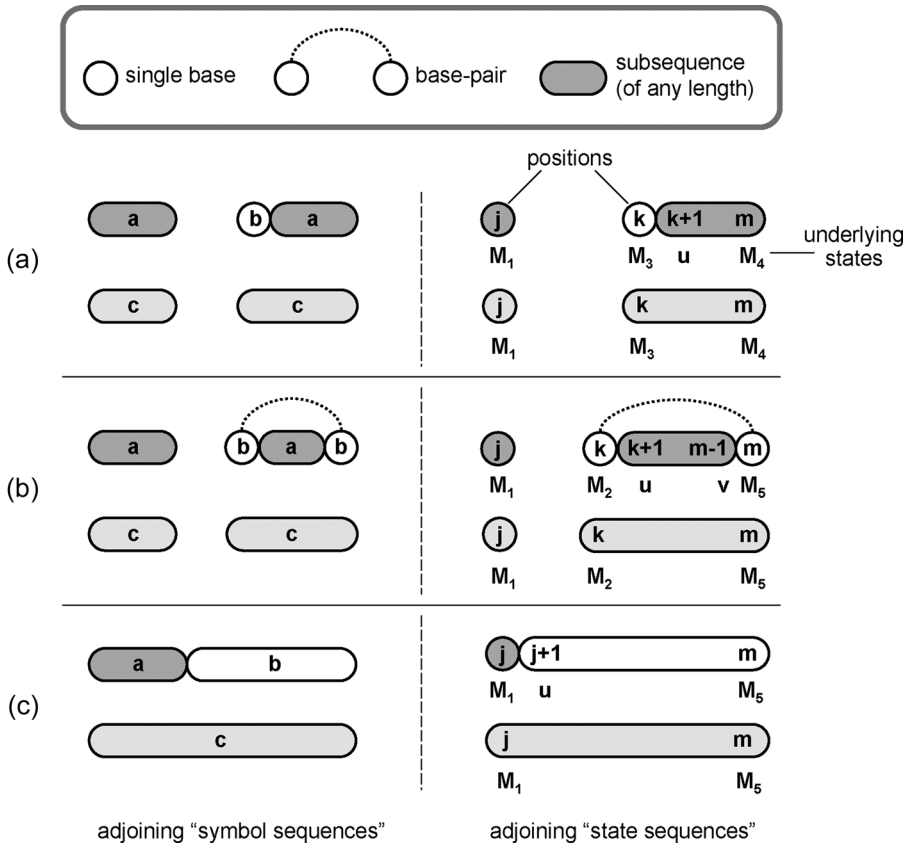


Fig. 3. Examples of adjoining rules.

sequence with two intervals (where the first interval has only one symbol at position j with underlying state M_1, and the second interval is located between $k+1$ and m whose left and right terminal states are u and M_4, respectively) and the *optimal state sequence with a single base* (located at k with underlying state M_3). When adjoining the two optimal state sequences, we consider *all* possible transitions from state M_3 at position k to state $u \in \{I_3, D_4, M_4\}$ at position $k+1$ and choose the one that maximizes the observation probability.[3] This exactly corresponds to STEP-2B shown on the right-hand side of Fig. 2. In a similar manner, the adjoining rule shown in Fig. 3(b) [or Fig. 3(c)] can be used to find the optimal sate sequence corresponding to the portion of the profile-csHMM shown in STEP-3B(or STEP-4B).

B. Handling Maximum Number of Base-Pairs

In principle, the adjoining rules that are used in the SCA algorithm are quite similar to the production rules (rewriting rules) in transformational grammars. In a transformational grammar, the production rules describe how a symbol sequence can be generated [1]. Therefore, the types of symbol correlations that can be represented by a grammar is determined by the set of production rules that is used. For example, context-free grammars have production rules that enable them to represent RNAs with nested base-pairs, but they cannot be used to represent RNAs with pseudoknots. In order to model pseudoknots, we need to use "context-sensitive" production rules that allow the reordering of certain symbols [1]. Note that adding such rules to a context-free grammar turns it into a context-sensitive grammar.

While the production rules are used for "generating" symbol sequences, the adjoining rules in the SCA algorithm are used for "tracing back" the most probable path by which a symbol sequence may have been generated by the given profile-csHMM [12]. In order to find the optimal path, the adjoining rules describe how we can find the optimal path of a given symbol sequence by optimally adjoining the optimal paths of its subsequences. For this reason, the types of symbol correlations in the profile-csHMM that can be traced back by the SCA algorithm is determined by the set of adjoining rules that are being considered.

Although it is true that including more complex adjoining rules in the SCA algorithm (or production rules in a transformational grammar) allows us to handle sequences with more complicated symbol correlations, the main reason for restricting the set of rules is to prevent the computational complexity for analyzing sequences from growing too large. Table I shows the computational complexity of the SCA algorithm for analyzing RNAs with different types of secondary structures, where L is the length of the sequence that is being analyzed.[4] As we can see in Table I, the computational cost for aligning RNAs increases with their structural complexity. For typical RNA pseudoknots,

[3]In this example, we can have $u = M_4$ only when $k+1 = m$.

[4]Actually, the computational complexity also depends on the number of states in the model. However, in this discussion, we consider it only in terms of the target sequence length L.

TABLE I
COMPUTATIONAL COMPLEXITY FOR ALIGNING RNAS WITH VARIOUS SECONDARY STRUCTURES

RNA Secondary Structure	Computational Complexity
Hairpin Structure	$O(L^2)$
tRNA Cloverleaf Structure	$O(L^3)$
Pseudoknots	$\geq O(L^4)$

the computational cost is $O(L^4)$, and for more complex pseudoknots in the Rivas and Eddy class [8], the cost can be as high as $O(L^6)$. Theoretically, the computational complexity can become even higher for more complex structures. Too high a complexity can make the computational analysis of RNA sequences practically infeasible—especially, when the sequence is long. Therefore, it is necessary to restrict the adjoining rules such that the maximum computational cost would not exceed the computational resource that is available.

Since profile-csHMMs can represent *any* kind of pairwise symbol correlations [12], it is possible to construct a profile-csHMM whose correlation structure is beyond the capacity of the present set of adjoining rules. In such cases, the SCA algorithm cannot properly find the optimal state sequence in the given profile-csHMM. Therefore, when building a profile-csHMM based on a consensus RNA sequence, it is important to ensure that its correlation structure is within the analytic capacity of the set of allowed adjoining rules.

Now, consider the case when we want to build a profile-csHMM for an RNA family, whose consensus sequence has a complicated secondary structure that cannot be handled by the current set of adjoining rules. In such cases, we need to "reduce" the original structure to a less complicated one that can be properly handled by the given set of rules. When finding a reduced secondary structure that can be analyzed by the present set of adjoining rules, it is important to find the one that includes the maximum number of base-pairs in the original structure. In this way, we can ensure that the profile-csHMM constructed based on the modified structure is the most accurate representation of the original RNA that is analyzable.

This idea is illustrated in Fig. 4. Let us first consider the RNA secondary structure shown in Fig. 4(a) that has multiple stems. Assume that we want to use only simple adjoining rules that do not allow the "bifurcation" of multiple stems so that the computational complexity does not exceed $O(L^2)$ [1]. In this case, we can remove a base-pair as shown in Fig. 4(a), in order to reduce the structure to a simpler one that can be handled by the given set of adjoining rules. It is not difficult to verify that the reduced structure shown in this figure is indeed the one that retains the maximum number of base-pairs. Fig. 4(b) shows another example, where the original RNA has a pseudoknot. Now, let us assume that we want to consider only nested base-pairs, since dealing with pseudoknots is computationally more expensive. In this case, we can remove the base-pair that crosses all the other base-pairs, to obtain the reduced structure shown in Fig. 4(b).

In these examples, it was relatively easy to find the reduced structures with the maximum number of base-pairs. However, finding the best substructure may not be straightforward in practice, where the RNAs are much longer and have more complex secondary structures. Furthermore, if we include diverse

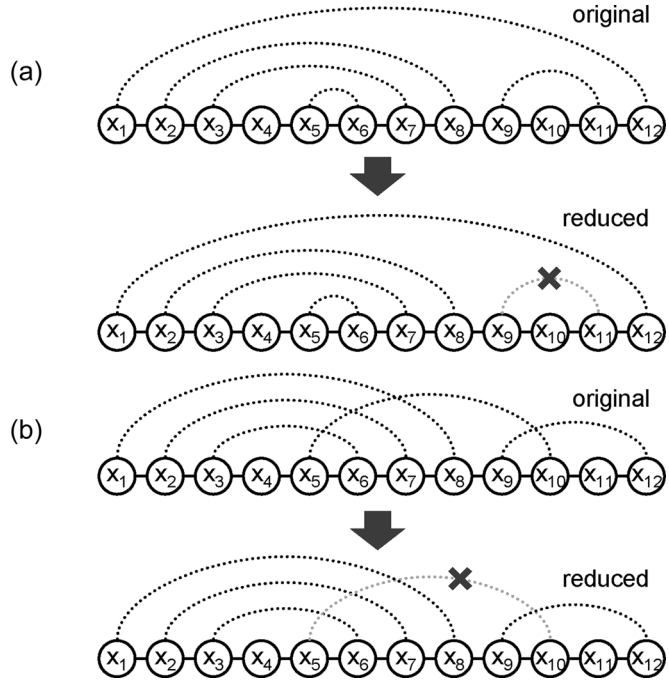


Fig. 4. Reducing the original RNA secondary structure to a simpler structure that can be handled by a given set of adjoining rules. (a) Reducing to a structure without bifurcating stems. (b) Reducing to a structure without pseudoknots.

adjoining rules to efficiently handle a large class of RNA secondary structures, it becomes also harder to determine whether a given secondary structure belongs to this class or not. Therefore, we need a systematic method for finding the best substructure of a given RNA that is analyzable by a specified set of adjoining rules.

C. Minimizing the Computational Cost for Finding the Optimal Alignment

As we mentioned earlier, the SCA algorithm has a variable computational complexity that depends on how the adjoining order is defined. For a profile-csHMM, there typically exist many legitimate adjoining orders, where any of them can be used to find the optimal state sequence. For example, let us consider the Viterbi algorithm that is used to find the optimal path in a traditional HMM. The algorithm tries to find the best alignment between the HMM and the symbol sequence starting from the leftmost position, and then it proceeds to the rightward direction until the optimal path of the entire sequence has been found. However, it is also possible to begin by aligning the rightmost parts of the model and the sequence, and proceed right-to-left to find the optimal path. In principle, both schemes can find the optimal path identically, unless there are multiple optimal paths that are equally probable. The computational complexity for finding the optimal path will be $O(L)$ for both schemes. Now, assume that we want to find the alignment between the symbol sequence and the HMM, starting from the inside of the model and proceeding towards outside, like the CYK algorithm that is used for parsing context-free grammars [1]. This also allows us to find the optimal state sequence, but the computational complexity will be increased to $O(L^2)$. This discussion clearly shows that there can be many ways for

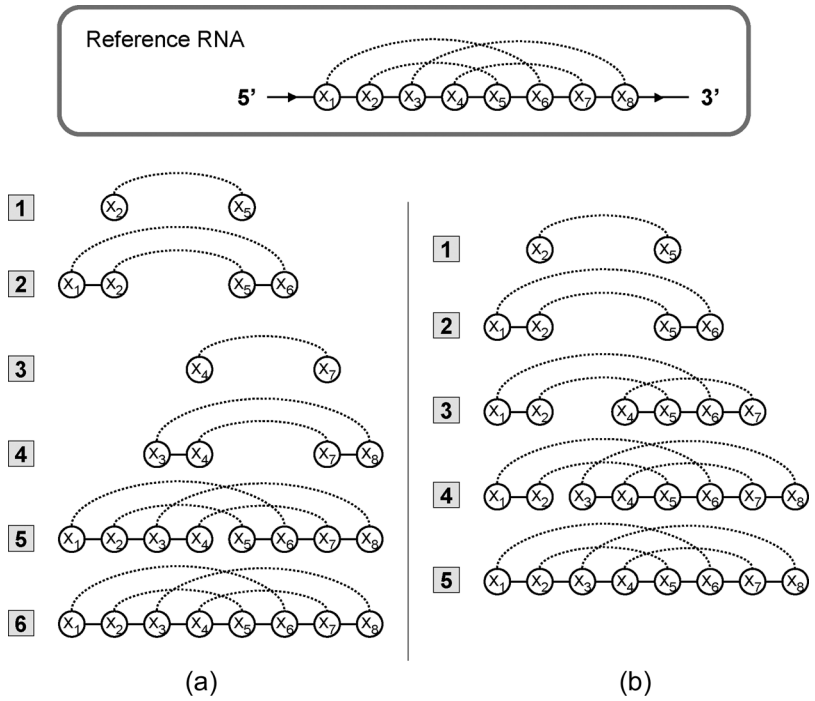


Fig. 5. Different adjoining orders may have different computational costs. (a) Adjoining order with a higher computational cost. (b) Adjoining order with a lower computational cost.

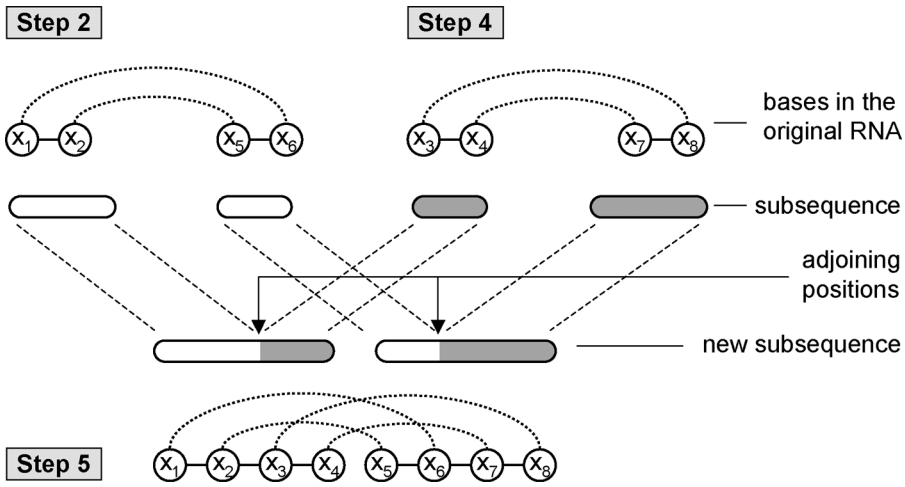


Fig. 6. Illustration of STEP-5 of the adjoining order in Fig. 5(a).

finding the optimal state sequence and that the computational cost for finding the optimal state sequence depends on how we proceed to find it.

As an example, let us assume that we want to construct a profile-csHMM based on the reference RNA shown in Fig. 5. Once the model is constructed, we can use it to find the best structural alignment between the reference RNA and other RNAs as elaborated in Section II. Since the given RNA has a pseudoknot structure, the constructed profile-csHMM displays crossing symbol correlations. Therefore, when finding the optimal alignment, we cannot simply proceed left-to-right (like the Viterbi algorithm) or inside-to-outside (like the CYK algorithm) but we have to follow a properly defined adjoining order. Fig. 5(a) and (b) shows two possible adjoining orders, where we can follow either of them to find the optimal state sequence in the profile-csHMM. However, the computational cost for following each order is significantly different, where it is $O(L^6)$ for the order shown in Fig. 5(a) and $O(L^4)$ for the order in Fig. 5(b).

The main reason for this large difference in computational cost is the following. When following the "high cost" adjoining order shown in Fig. 5(a), the SCA algorithm tries to optimally adjoin two subsequences, where both subsequences have two intervals [see STEP-5 in Fig. 5(a)]. This is illustrated in Fig. 6, where the corresponding bases in the original RNA are shown along with each subsequence. In order to adjoin the sequences in an optimal way, the algorithm has to consider all possible po-

sitions for adjoining the intervals and choose the positions that maximize the observation probability of the new subsequence. The computational cost of this process is very high (i.e, $O(L^6)$), making STEP-5 the bottleneck of the entire adjoining process. Unlike this "high cost" order in Fig. 5(a), the "low cost" adjoining order shown in Fig. 5(b) tries to extend the intermediate subsequences either by a single base or a single base-pair at a time. This eliminates the need for choosing the optimal adjoining positions as in Fig. 6, hence the overall complexity is reduced to $O(L^4)$.

In the example shown in Fig. 5, it may not be difficult to find out which of the two schemes has a lower cost, as their computational complexities have different orders. However, the problem of choosing the adjoining order with the lowest computational cost is more difficult and subtle, as there typically exist many adjoining orders whose complexities have the same order. Therefore, we definitely need an effective method for finding the optimal adjoining order that minimizes the computational cost, or equivalently, that makes the structural alignment speed the fastest.

IV. Finding the Optimal Adjoining Order

In Section III, we have discussed two important things that have to be considered in finding the adjoining order, in order to make structural RNA alignments based on profile-csHMMs faster and more accurate. In this section, we propose an efficient algorithm that can be used for finding the "optimal" adjoining order, based on the criterion considered in Section III. More precisely, the proposed algorithm solves the following problem.

Problem Statement:

Given a reference RNA sequence with structural annotation:

1) if necessary, find the substructure of the original RNA that contains the maximum number of base-pairs that can be handled by the current set of adjoining rules;
2) find the adjoining order (for the profile-csHMM that represents the reference RNA or the modified RNA if a substructure was chosen) that minimizes the computational complexity of the SCA algorithm.

Even though we have described the purpose of the proposed algorithm in two separate steps for clarity, *finding the best substructure* and *finding the optimal adjoining order* actually proceed simultaneously, as we will see in Section IV-C. Note that if the original RNA can be handled by the current set of adjoining rules, the algorithm directly uses the original RNA since there is no need to modify the structure of the RNA to a simpler one.

A. Notation

In order to describe the algorithm for finding the optimal adjoining order, let us first define the necessary notation. The reference RNA sequence is denoted by $\mathbf{x} = x_1x_2x_3 \dots x_K$, where K is the length of the RNA. In order to describe a subsequence of $\mathbf{x}$, we use an ordered set of one or more nonoverlapping closed intervals $\mathcal{N} = \{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_I\}$. The ith interval is denoted by $\mathbf{n}_i = [n_i^\ell, n_i^r]$, where n_i^ℓ is the left end of the ith interval and n_i^r is its right end. Every interval $\mathbf{n}_i = [n_i^\ell, n_i^r]$ in the set $\mathcal{N}$ satisfies

$$1 \le n_i^\ell \le n_i^r \le K$$

and

$$n_i^r < n_j^\ell \text{ for } i < j.$$

Using the set $\mathcal{N}$, we can define a subsequence of $\mathbf{x}$ as follows:

$$\mathbf{x}(\mathcal{N}) = \underbrace{x_{n_1^\ell} \dots x_{n_1^r}}_{\mathbf{n}_1} \underbrace{x_{n_2^\ell} \dots x_{n_2^r}}_{\mathbf{n}_2} \dots \underbrace{x_{n_I^\ell} \dots x_{n_I^r}}_{\mathbf{n}_I}.$$

For notational convenience, we denote the length of this subsequence by $|\mathbf{x}(\mathcal{N})|$, or equivalently by $|\mathcal{N}|$, which is defined as follows:

$$|\mathbf{x}(\mathcal{N})| = |\mathcal{N}| = \sum_{i=1}^{I} \left(n_i^r - n_i^\ell + 1\right).$$

Based on this notation, we can define the following subsequences:

$$\mathcal{N}_1 = \{[2,2]\} \longrightarrow \mathbf{x}(\mathcal{N}_1) = x_2$$
$$\mathcal{N}_2 = \{[1,2],[4,4]\} \longrightarrow \mathbf{x}(\mathcal{N}_2) = x_1x_2\ x_4$$
$$\mathcal{N}_3 = \{[1,1],[3,5],[7,8]\} \longrightarrow \mathbf{x}(\mathcal{N}_3) = x_1\ x_3x_4x_5\ x_7x_8.$$

As we can see in these examples, using the set notation allows us to define subsequences with multiple intervals, which is indispensable for dealing with complicated base correlations in RNA sequences. One thing that should be noted is that there can be multiple sets that designate subsequences that are virtually identical. Such an example is shown in the following:

$$\mathcal{N}_4^a = \{[2,5]\} \longrightarrow \mathbf{x}\left(\mathcal{N}_4^a\right) = x_2x_3x_4x_5$$
$$\mathcal{N}_4^b = \{[2,3],[4,5]\} \longrightarrow \mathbf{x}\left(\mathcal{N}_4^b\right) = x_2x_3\ x_4x_5.$$

The only difference between these two subsequences is that $\mathbf{x}(\mathcal{N}_4^a)$ treats "$x_2x_3x_4x_5$" as one contiguous sequence, whereas $\mathbf{x}(\mathcal{N}_4^b)$ regards it as a concatenation of two sequences "x_2x_3" and "x_4x_5."

For any subsequence $\mathbf{x}(\mathcal{N})$, there typically exist many ways to obtain $\mathbf{x}(\mathcal{N})$ by adjoining two shorter subsequences $\mathbf{x}(\mathcal{N}_a)$ and $\mathbf{x}(\mathcal{N}_b)$ using one of the allowed adjoining rules. Given a subsequence $\mathbf{x}(\mathcal{N})$, we define the set of all such pairs $(\mathcal{N}_a, \mathcal{N}_b)$ as follows:

$$\mathcal{A}(\mathcal{N}) = \{(\mathcal{N}_a, \mathcal{N}_b) \mid \mathbf{x}(\mathcal{N}) = \mathbf{x}(\mathcal{N}_a) + \mathbf{x}(\mathcal{N}_b), \mathcal{N}_a < \mathcal{N}_b\}.$$

In this definition

$$\mathbf{x}(\mathcal{N}) = \mathbf{x}(\mathcal{N}_a) + \mathbf{x}(\mathcal{N}_b)$$

implies that the subsequence $\mathbf{x}(\mathcal{N})$ can be obtained by adjoining the two subsequences $\mathbf{x}(\mathcal{N}_a)$ and $\mathbf{x}(\mathcal{N}_b)$ using one of the adjoining rules. The second condition

$$\mathcal{N}_a < \mathcal{N}_b$$

implies that the leftmost interval of $\mathcal{N}_a$ is located on the left-hand side of the leftmost interval of $\mathcal{N}_b$. For example, the following inequalities hold using this notation:

$$\mathcal{N}_5^a = \{[1,3]\} < \mathcal{N}_5^b = \{[5,9]\}$$
$$\mathcal{N}_6^a = \{[2,3],[9,10]\} < \mathcal{N}_6^b = \{[4,5],[7,8]\}.$$

For a subsequence $\mathbf{x}(\mathcal{N})$, we define $n(\mathcal{N})$ as the **maximum number of base-pairs** in the subsequence $\mathbf{x}(\mathcal{N})$ that can be handled by the set of allowed adjoining rules. Now, assume that there are multiple ways—i.e, multiple adjoining orders—for applying the adjoining rules that can handle $n(\mathcal{N})$ base-pairs. In such cases, we want to choose the adjoining order that minimizes the computational cost for finding the best alignment between the subsequence $\mathbf{x}(\mathcal{N})$ of the reference RNA and a subsequence of the target RNA under consideration. For this purpose, we define $c(\mathcal{N})$ to be the **minimum computational cost** for applying the adjoining rules for finding this alignment. In order to compute $c(\mathcal{N})$, we need to define the initialization cost $c_{\text{init}}(\mathcal{N})$ and the adjoining cost $c_{\text{adj}}(\mathcal{N};\mathcal{N}_a,\mathcal{N}_b)$. The initialization cost $c_{\text{init}}(\mathcal{N})$ is the computational cost for finding the best alignment between a single base (or a single base-pair) in the reference RNA and a base (or a base-pair) in the target RNA, allowing gaps. The adjoining cost $c_{\text{adj}}(\mathcal{N};\mathcal{N}_a,\mathcal{N}_b)$ is the computational cost for adjoining two sequences $\mathbf{x}(\mathcal{N}_a)$ and $\mathbf{x}(\mathcal{N}_b)$ to obtain $\mathbf{x}(\mathcal{N})$. Finally, we define two variables $\lambda_a(\mathcal{N})$ and $\lambda_b(\mathcal{N})$ that are used to trace-back the optimal adjoining order that maximizes the number of considered base-pairs and minimizes the overall computational cost.

B. Initialization

We begin by initializing $n(\mathcal{N})$ and $c(\mathcal{N})$ for all subsequences $\mathbf{x}(\mathcal{N})$ that consists of either a single base or a single base-pair.

i) For a position k ($1 \le k \le K$), we let $\mathcal{N} = \{[k,k]\}$, and initialize $n(\mathcal{N})$ and $c(\mathcal{N})$ as follows:

$$\begin{aligned} n(\mathcal{N}) &= 0 \\ c(\mathcal{N}) &= c_{\text{init}}(\mathcal{N}) \\ \lambda_a(\mathcal{N}) &= \varnothing \\ \lambda_b(\mathcal{N}) &= \varnothing. \end{aligned}$$

ii) For positions j and k ($1 \le j < k \le K$), we let $\mathcal{N} = \{[j,j],[k,k]\}$, and initialize $n(\mathcal{N})$ and $c(\mathcal{N})$ as follows:

$$\begin{aligned} n(\mathcal{N}) &= \delta(j,k) \\ c(\mathcal{N}) &= c_{\text{init}}(\mathcal{N}) \\ \lambda_a(\mathcal{N}) &= \varnothing \\ \lambda_b(\mathcal{N}) &= \varnothing \end{aligned}$$

where

$$\delta(j,k) = \begin{cases} 1, & \text{if } x_j \text{ and } x_k \text{ form a base-pair} \\ 0, & \text{otherwise.} \end{cases}$$

C. Iteration

Now, we proceed to compute $n(\mathcal{N})$ and $c(\mathcal{N})$ for longer subsequences in an iterative manner. We begin by computing these values for all subsequences of length $|\mathcal{N}| = 2$, and continue the iterative process until we obtain $n(\mathcal{N})$ and $c(\mathcal{N})$ for $\mathcal{N} = \{[1,K]\}$, which corresponds to the complete sequence $\mathbf{x} = x_1 x_2 \ldots x_K$. For each subsequence $\mathbf{x}(\mathcal{N})$, we take the following steps to compute $n(\mathcal{N})$ and $c(\mathcal{N})$.

i) First, we find the set $\mathcal{A}(\mathcal{N})$ of all possible partitions $(\mathcal{N}_a,\mathcal{N}_b)$ for the given subsequence.

ii) Second, we compute

$$n(\mathcal{N}) = \max_{(\mathcal{N}_a,\mathcal{N}_b)\in\mathcal{A}(\mathcal{N})} [n(\mathcal{N}_a) + n(\mathcal{N}_b)].$$

In this step, we can compute the maximum number of base-pairs in the subsequence $\mathbf{x}(\mathcal{N})$ that can be handled by the given set of adjoining rules.

iii) Third, we find the subset of $\mathcal{A}(\mathcal{N})$ that consists of all partitions $(\mathcal{N}_a,\mathcal{N}_b)$ that can achieve the maximum number of base-pairs $n(\mathcal{N})$. This subset $\mathcal{A}^*(\mathcal{N})$ is defined as

$$\mathcal{A}^*(\mathcal{N}) = \{(\mathcal{N}_a,\mathcal{N}_b) \mid (\mathcal{N}_a,\mathcal{N}_b) \in \mathcal{A}(\mathcal{N}) \quad n(\mathcal{N}_a) + n(\mathcal{N}_b) = n(\mathcal{N})\}.$$

Among all the partitions in $\mathcal{A}^*(\mathcal{N})$, now we find the one that minimizes the computational cost

$$\begin{aligned} c(\mathcal{N}) &= \min_{(\mathcal{N}_a,\mathcal{N}_b)\in\mathcal{A}^*(\mathcal{N})} [c(\mathcal{N}_a) + c(\mathcal{N}_b) \\ &\qquad + c_{\text{adj}}(\mathcal{N};\mathcal{N}_a,\mathcal{N}_b)] \\ (\mathcal{N}_a^*,\mathcal{N}_b^*) &= \arg\min_{(\mathcal{N}_a,\mathcal{N}_b)\in\mathcal{A}^*(\mathcal{N})} [c(\mathcal{N}_a) + c(\mathcal{N}_b) \\ &\qquad + c_{\text{adj}}(\mathcal{N};\mathcal{N}_a,\mathcal{N}_b)] \\ \lambda_a(\mathcal{N}) &= \mathcal{N}_a^* \\ \lambda_b(\mathcal{N}) &= \mathcal{N}_b^*. \end{aligned}$$

D. Trace-Back

Once we have computed $n(\mathcal{N})$ and $c(\mathcal{N})$ for $\mathcal{N} = \{[1,K]\}$, we trace-back the algorithm to find the optimal adjoining order. For this purpose, we need two empty stacks S and T, where T will be used to store the intermediate results and S will be used to store the optimal adjoining order. The respective adjoining instruction is denoted by $(\mathcal{N} : \mathcal{N}_a,\mathcal{N}_b)$, which indicates that we can obtain the subsequence $\mathbf{x}(\mathcal{N})$ by adjoining $\mathbf{x}(\mathcal{N}_a)$ and $\mathbf{x}(\mathcal{N}_b)$. Note that this adjoining instruction implies which adjoining rule should be used.

1) Empty S and T. Push $\mathcal{N} = \{[1,K]\}$ onto the stack T.
2) Pop $\mathcal{N}^t$ from T. Let $\mathcal{N}_a^t = \lambda_a(\mathcal{N}^t)$, $\mathcal{N}_b^t = \lambda_b(\mathcal{N}^t)$.
3) Push $(\mathcal{N}^t : \mathcal{N}_a^t,\mathcal{N}_n^t)$ onto the stack S.
4) If $\mathcal{N}_a^t \neq \varnothing$, push $\mathcal{N}_a^t$ onto T.
5) If $\mathcal{N}_b^t \neq \varnothing$, push $\mathcal{N}_b^t$ onto T.
6) If T is empty, terminate. Otherwise, goto STEP 2.

After the termination of the above procedure, the stack S contains the optimal adjoining order.

E. Example

As an example, let us again consider the RNA sequence shown in Fig. 1(a). The dotted lines indicate the base-pairs. For this reference RNA, assume that we have obtained the following adjoining order using the proposed algorithm.

1) ({[1,1],[4,4]} :∅ ,∅).

2) ({[1,1],[3,4]} :{[1,1],[4,4]} ,{[3,3]}).

3) ({[1,1],[2,5]} :{[1,1],[3,4]} , {[2,2],[5,5]}).

4) ({[1,5]} :{[1,1],[2,5]} ,∅).

The number on the left-hand side indicate the order of each adjoining instruction in the stack S, counted from top to bottom. This adjoining order corresponds to the step-by-step "assembly order" of the original RNA, as shown in Fig. 2. Note that we may have $\mathcal{N}_a = \emptyset$ or $\mathcal{N}_b = \emptyset$ in an adjoining instruction $(\mathcal{N} : \mathcal{N}_a, \mathcal{N}_b)$. If $\mathcal{N}_a = \mathcal{N}_b = \emptyset$, this corresponds to an initialization rule for a stand-alone base or a stand-alone base-pair. Such a case is shown in STEP 1 in Fig. 2. If only one of them is an empty set, let us say $\mathcal{N}_a \neq \emptyset$ and $\mathcal{N}_b = \emptyset$, then the instruction implies that one or more intervals in $\mathcal{N}_a$ are combined such that $\mathcal{N}$ has fewer intervals than $\mathcal{N}_a$. For example, the last instruction

$$(\{[1,5]\} : \{[1,1],[2,5]\}, \emptyset)$$

shows that the two intervals [1, 1] and [2, 5] in $\mathcal{N}_a$ are combined into one interval, such that $\mathcal{N}$ has only one interval, which is [1, 5]. This is illustrated in STEP 4 in Fig. 2.

F. Estimating the Computational Costs for Adjoining Rules

In order to carry out the proposed optimization algorithm, we first have to estimate the computational costs $c_{\text{init}}(\mathcal{N})$ for initialization and the costs $c_{\text{adj}}(\mathcal{N}; \mathcal{N}_a, \mathcal{N}_b)$ for adjoining subsequences. Although the exact costs depend on various factors—such as how the SCA algorithm is implemented, which programming language is used, what kind of compiler is used to build the program, on which machine the algorithm is executed, and so on—a reasonable way for estimating these costs is to count the number of iterations that the algorithm has to go through in order to carry out the corresponding initialization or adjoining rule. One small problem with this approach is that the number of iterations depend on the length of the target RNA sequence, which is generally not known in advance. In such cases, we can simply use the average length of the RNAs in the reference RNA family (or equivalently, the length of the consensus RNA sequence) used to construct the profile-csHMM. This will yield the adjoining order that will be optimal for the average case.

V. Experimental Results

In order to demonstrate the proposed idea and to verify how much performance gain can be obtained by using the optimal adjoining order, we applied the algorithm described in Section IV to the profile-csHMM based structural alignment method proposed in [15]. The new structural alignment method that employs the optimized adjoining order has been tested using several RNA sequence families in the Rfam database [5]. We compared its performance with the original implementation [15] as well as the PSTAG-based method [6], which is a state-of-the-art structural alignment method that can handle many pseudoknots.

For our experiments, we chose the following RNA families: CORONA_PK3, HDV_RIBOZYME, TOMBUS_3_IV, and FLAVI_PK3. Note that all these families contain pseudoknots, which cannot be handled by traditional SCFGs. The reason for choosing these pseudoknotted families was as follows. For an RNA that does not have a pseudoknot, the SCA algorithm can simply proceed "inside-to-outside" like the CYK algorithm for SCFGs. In this case, the adjoining order will be more or less fixed, and there will not be a significant difference between an optimal order and a suboptimal one. However, for an RNA with a pseudoknot, the computational complexity of the SCA algorithm may differ significantly depending on how we define the adjoining order. One such example was shown in Fig. 5. For this reason, we chose RNA families with pseudoknots as they will clearly demonstrate the importance of using the optimal adjoining order with minimum complexity. In our experiments, we used the RNA sequences in the "seed" alignment of each RNA family, as they have a relatively reliable structural annotation. Table IV summarizes the basic properties of the four RNA families, such as the number of RNAs in the seed alignment, the average length of the member sequences, and their average percentage (sequence) identity.[5]

The experiments have been performed as follows. For a given RNA sequence family, we first chose a member of this family and used it as the reference RNA to build a profile-csHMM. Then we found the optimal adjoining order by using the proposed algorithm. Following this optimal order, we applied the SCA algorithm to find the best structural alignment between the reference RNA and each of the remaining members in the same family. This process has been repeated for all the members in the given family, to estimate the average performance of the alignment method. In order to evaluate the quality of the alignments, we predicted the secondary structure of the target RNA based on the respective alignment and compared it to the annotated structure in the database. Then we counted the number of correctly predicted base-pairs (TP; true positives), the number of incorrectly predicted base-pairs (FP; false positives), and the number of base-pairs that are present in the annotated structure but were not predicted by the proposed method (FN; false negatives). These numbers were used to compute the average *sensitivity* (SN) and *specificity* (SP) of the prediction, which are defined as

$$\text{SN} = \text{TP}/(\text{TP} + \text{FN}), \quad \text{SP} = \text{TP}/(\text{TP} + \text{FP}).$$

The results of these cross-validation experiments are summarized in Table II and Table III. Let us first take a look at Table II, which shows the prediction accuracy of the structural alignment methods. For convenience, the highest prediction ratios have been boldfaced. First, we can see that the prediction performance of the profile-csHMM approach and that of the

[5]These numbers have been obtained from the Rfam database (version 8.1). FLAVI_PK3 seed sequences were obtained from Rfam 7.0, which are included in the FLAVI_CRE family in Rfam 8.1.

TABLE II
PREDICTION ACCURACY OF THE STRUCTURAL ALIGNMENT METHODS

	PROFILE-CSHMM				PSTAG	
	OPTIMAL		ORIGINAL			
	SN (%)	SP (%)	SN (%)	SP (%)	SN (%)	SP (%)
CORONA_PK3	**95.7**	**96.5**	**95.7**	**96.5**	94.6	95.5
HDV_RIBOZYME	**94.5**	95.3	**94.5**	95.3	94.1	**95.6**
TOMBUS_3_IV	**98.6**	**98.6**	96.9	96.9	97.4	97.4
FLAVI_PK3	**94.5**	**96.4**	**94.5**	**96.4**	-	-

TABLE III
AVERAGE CPU TIME FOR FINDING AN RNA STRUCTURAL ALIGNMENT

	PROFILE-CSHMM			PSTAG
	OPTIMAL	ORIGINAL	ORG / OPT	
	sec	sec	ratio	sec
CORONA_PK3	0.68	1.2	1.7	37.2
HDV_RIBOZYME	0.58	1.7	2.9	207.5
TOMBUS_3_IV	0.42	1.0	2.4	270.9
FLAVI_PK3	1.87	6.8	3.6	-

TABLE IV
BASIC PROPERTIES OF THE RNA FAMILIES

	# of seed sequences	average length	average percentage identity
CORONA_PK3	14	62.5	70
HDV_RIBOZYME	15	88.8	95
TOMBUS_3_IV	18	64.5	94
FLAVI_PK3	14	95.4	69

PSTAG approach are comparable to each other for the first three RNA families. Both methods achieved considerably high prediction accuracies, indicating that these methods are capable of finding accurate pairwise alignment of RNAs with pseudoknots. However, as we mentioned earlier, profile-csHMMs can also handle many RNAs whose structure is too complex to be handled by PSTAGs.[6] The FLAVI_PK3 RNA family is one such example. The results in Table II show that the profile-csHMM approach can find good alignments also for these relatively complex RNAs. Second, we can observe that the profile-csHMM based method shows nearly identical prediction performances for the two different adjoining orders, as we expect. The column labeled as "optimal" shows the performance when using the optimal adjoining order that is obtained using the algorithm proposed in Section IV, while the column "original" shows the performance for using the adjoining order that is found by the original implementation of the profile-csHMM alignment method [15]. Interestingly, for TOMBUS_3_IV, there were cases when a number of different alignments received identical alignment scores (i.e., observation probabilities), hence the two adjoining orders resulted in different predictions.

Table III shows the average CPU time for finding a structural alignment using the respective method.[7] As we can see in this table, the use of an optimized adjoining order made the alignment speed up to 3.6 times faster. Considering that the "original" adjoining order obtained using the original method [15] is already a fairly good one, this improvement is quite interesting. In the original method, the alignment program tries to avoid high-cost adjoining rules and use low-cost adjoining rules whenever possible. Therefore, even though the original method is not able to find the optimal adjoining order, it yields a reasonably good order. In fact, the computational complexity of the SCA algorithm was $O(L^4)$ for both types of adjoining orders, for every RNA that was used in our experiments. This shows that even for adjoining orders with the same order of algorithmic complexity, there can be a considerable difference between their actual computational costs. Another interesting observation is that the average speed improvement that is obtained by optimizing the adjoining order is greater for more complex RNAs. This makes intuitive sense, since for more complex structures, it would be less likely that a randomly chosen adjoining order would be close to the optimum one. For comparison, Table III also shows the alignment speed of the PSTAG approach, where we can see that the profile-csHMM based method runs significantly faster despite its larger descriptive capability.

[6]Theoretically, profile-csHMMs can handle RNA secondary structures with any kind of base-pairs. In our current implementation, the profile-csHMM based structural alignment method can handle any RNA that belongs to the Rivas&Eddy class [8].

[7]All experiments have been performed on a PowerMac G5 2.5 GHz with 4-GB memory.

VI. CONCLUDING REMARKS

In this paper, we have proposed an efficient algorithm that can be used for finding the optimal adjoining order of the SCA algorithm. As elaborated in Sections III and IV, the proposed algorithm achieves two important goals. First, if the RNA of interest has a complicated structure that cannot be handled by the current set of adjoining rules, the algorithm can find the optimal substructure with the maximum number of base-pairs that can be handled by the adjoining rules at hand. Second, for an RNA secondary structure that is within the analytic capability of the present set of adjoining rules, the algorithm gives us the adjoining order that minimizes the computational cost for finding the structural alignment between the given reference RNA and other target RNAs. Experimental results show that the proposed algorithm can make the average alignment speed up to 3.6 times faster without any degradation in the quality of the alignments. As mentioned earlier, the adjoining rules that are used to traceback the optimal state sequence in a profile-csHMM are similar to the production rules in a transformational grammar that are used to generate symbol sequences. Therefore, we believe that a similar approach can be used for finding the optimal tree structure of other context-sensitive grammars that minimizes the computational cost for parsing the tree. This is an interesting topic for further investigation.

REFERENCES

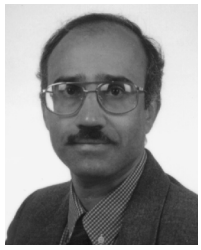
[1] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[2] S. R. Eddy and R. Durbin, "RNA sequence analysis using covariance models," *Nucl. Acids Res.*, vol. 22, pp. 2079–2088, 1994.
[3] S. R. Eddy, "Non-coding RNA genes and the modern RNA world," *Nature Rev. Genetics*, vol. 2, pp. 919–929, 2001.
[4] O. Gotoh, "An improved algorithm for matching biological sequences," *J. Mol. Biol.*, vol. 162, pp. 705–708.
[5] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy, "Rfam: An RNA family database," *Nucl. Acids Res.*, vol. 31, pp. 439–441, 2003.
[6] H. Matsui, K. Sato, and Y. Sakakibara, "Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures," *Bioinformatics*, vol. 21, pp. 2611–2617, 2005.

[7] J. S. Mattick, "Challenging the dogma: The hidden layer of non-protein-coding RNAs in complex organisms," *BioEssays*, vol. 25, pp. 930–939, 2003.
[8] E. Rivas and S. R. Eddy, "The language of RNA: A formal grammar that includes pseudoknots," *Bioinformatics*, vol. 16, pp. 334–340, 2000.
[9] Y. Sakakibara, "Pair hidden Markov models on tree structures," *Bioinformatics*, vol. 19, pp. i232–i240, 2003.
[10] G. Storz, "An expanding universe of noncoding RNAs," *Science*, vol. 296, pp. 1260–1263, 2002.
[11] B.-J. Yoon and P. P. Vaidyanathan, "Context-sensitive hidden Markov models for modeling long-range dependencies in symbol sequences," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4169–4184, Nov. 2006.
[12] B.-J. Yoon and P. P. Vaidyanathan, "Profile context-sensitive HMMs for probabilistic modeling of sequences with complex correlations," presented at the 31st Int. Conf. Acoustics, Speech, and Signal Processing, Toulouse, May 2006.
[13] B.-J. Yoon and P. P. Vaidyanathan, "Computational identification and analysis of noncoding RNAs—Unearthing the buried treasures in the genome," *IEEE Signal Process. Mag.*, vol. 24, no. 1, pp. 64–74, Jan. 2007.
[14] B.-J. Yoon and P. P. Vaidyanathan, "Fast search of sequences with complex symbol correlations using profile context-sensitive HMMs and pre-screening filters," presented at the Proc. 32nd Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP), Honolulu, HI, Apr. 2007.
[15] B.-J. Yoon and P. P. Vaidyanathan, "Structural alignment of RNAs using profile-csHMMs and its application to RNA homology search: Overview and new results," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 10–25, Jan. 2008, Joint Special Issue on Systems Biology with *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*.

Byung-Jun Yoon (S'02–M'07) was born in Seoul, Korea, in 1975. He received the B.S.E. (summa cum laude) degree from Seoul National University (SNU), Seoul, in 1998, and the M.S. and Ph.D. degrees from the California Institute of Technology (Caltech), Pasadena, in 2002 and 2007, respectively, all in electrical engineering.

He was a Postdoctoral Researcher at Caltech from December 2006 to October 2007. In 2008, he joined the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, where he is currently an Assistant Professor. His main research interest is in bioinformatics, genomic signal processing, and systems biology.

Dr. Yoon received the Killgore Fellowship in 2001 from Caltech, and he was selected as a recipient of the Microsoft Research Graduate Fellowship for the year 2004–2005. In 2003, he was awarded a prize in the student paper contest in the 37th Asilomar conference on signals, systems, and computers.



P. P. Vaidyanathan (S'80–M'83–SM'88–F'91) was born in Calcutta, India, on October 16, 1954. He received the B.Sc. (Hons.) degree in physics and the B.Tech. and M.Tech. degrees in radio physics and electronics, all from the University of Calcutta, in 1974, 1977, and 1979, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1982.

He was a Postdoctoral Fellow at the University of California, Santa Barbara, from September 1982 to March 1983. In March 1983, he joined the Electrical Engineering Department, California Institute of Technology, Pasadena, as an Assistant Professor, and since 1993 has been Professor of electrical engineering. His main research interests are in digital signal processing, multirate systems, wavelet transforms, and signal processing for digital communications. He has authored a number of papers in IEEE journals, and is the author of the book *Multirate Systems and Filter Banks* (Englewood Cliffs, NJ: Prentice-Hall, 1993). He has written several chapters for various signal processing handbooks.

Dr. Vaidyanathan served as Vice-Chairman of the Technical Program Committee for the 1983 IEEE International Symposium on Circuits and Systems and Technical Program Chairman for the 1992 IEEE International symposium on Circuits and Systems. He was an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS during 1985–1987, and is currently an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS and a consulting editor for the *Applied and Computational Harmonic Analysis* journal. He was a Guest Editor in 1998 for special issues of the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, on the topics of filter banks, wavelets, and subband coders. He was a recipient of the Award for Excellence in Teaching at the California Institute of Technology for the years 1983–1984, 1992–1993, and 1993–1994. He also received the NSF's Presidential Young Investigator award in 1986. In 1989, he received the IEEE ASSP Senior Award for his paper on multirate perfect-reconstruction filter banks. In 1990, he was the recipient of the S. K. Mitra Memorial Award from the Institute of Electronics and Telecommunications Engineers, India, for his joint paper in the *IETE Journal*. He was also the coauthor of a paper on linear-phase perfect reconstruction filter banks in the IEEE TRANSACTIONS ON SIGNAL PROCESSING, for which the first author (T. Nguyen) received the Young Outstanding Author award in 1993. He received the 1995 F. E. Terman Award of the American Society for Engineering Education, sponsored by Hewlett Packard Co. for his contributions to engineering education. He has given several plenary talks including at the Sampta'01, Eusipco'98, SPCOM'95, and Asilomar'88 conferences on signal processing. He has been chosen a distinguished lecturer for the IEEE Signal Processing Society for the year 1996–1997. In 1999, he received the IEEE CAS Society's Golden Jubilee Medal. He is a recipient of the IEEE Signal Processing Society's Technical Achievement Award for 2002.